$f \circ g = h \circ f$

Speculative Deforestation

William Sonnex

UNIVERSITY OF CAMBRIDGE

1. Introduction

- Deforestation is a function simplification technique invented by Philip Wadler
- It optimises functional programs by removing intermediary results e.g. $map\ g\ (map\ f\ xs) \leadsto map\ (g \circ f)\ xs$
- We have developed extensions to yield simpler results, not for runtime, but for program verification and ATP
- These extensions we have collectively named "factoring", as they factor out a context from a recursive function, i.e. $\mu h \leadsto f(\mu g)$, factoring f out of μh to yield μg
- In this poster we present only "constant context" factoring

2. Deforesting $rev\ (xs \mathbin{+\!\!+} [n])$

Fix a new $rev_{+\!\!+} : [a] \to a \to [a]$, s.t.

$$rev_{+\!\!+}\ xs\ n \triangleq rev\ (xs \mathbin{+\!\!+} [n])$$

Deforest the definition of rev_2:

$$rev_{+\!\!+}\ xs\ n = T[\![rev\ (xs \mathbin{+\!\!+} [n])]\!]$$

Expand ($+\!\!+$)

$$=_\beta T\left[\!\!\left[rev\ \left(xs\begin{cases}[\,] \to [n] \\ (c :: cs) \to c :: (cs \mathbin{+\!\!+} [n])\end{cases}\right)\right]\!\!\right]$$

$T[\![rev\ (\_)]\!]$ floats inside

$$= xs\begin{cases}[\,] \to T[\![rev\ [n]]\!] \\ (c :: cs) \to T[\![rev\ (c :: (cs \mathbin{+\!\!+} [n]))]\!]\end{cases}$$

Expand rev

$$=_\beta xs\begin{cases}[\,] \to T[\![[n]]\!] \\ (c :: cs) \to T[\![rev\ (cs \mathbin{+\!\!+} [n]) \mathbin{+\!\!+} [c]]\!]\end{cases}$$

Apply definition $rev_{+\!\!+}\ xs\ n \triangleq rev\ (xs \mathbin{+\!\!+} [n])$

$$= xs\begin{cases}[\,] \to T[\![[n]]\!] \\ (c :: cs) \to T[\![(rev_{+\!\!+}\ xs\ n) \mathbin{+\!\!+} [c]]\!]\end{cases}$$

Finish

$$= xs\begin{cases}[\,] \to [n] \\ (c :: cs) \to (rev_{+\!\!+}\ xs\ n) \mathbin{+\!\!+} [c]\end{cases}$$

Optimized recursive definition for $rev_{+\!\!+}\ xs\ n$

3. Factoring $rev_{+\!\!+}\ xs\ n$ into $n :: (rev\ xs)$

Deforestation found $rev\ (xs \mathbin{+\!\!+} [n]) \leadsto rev_{+\!\!+}\ xs\ n$

Constant context factoring finds $rev_{+\!\!+}\ xs\ n \leadsto n :: (rev\ xs)$

First, speculate the constant context C using a dynamic approach, enumerate inputs to $rev_{+\!\!+}\ xs\ n$:

$$(rev_{+\!\!+}\ xs\ n)[xs \mapsto [\,]] =_\beta [n]$$
$$(rev_{+\!\!+}\ xs\ n)[xs \mapsto [a]] =_\beta [n, a]$$
$$(rev_{+\!\!+}\ xs\ n)[xs \mapsto [a,b]] =_\beta [n, b, a]$$

All are $n :: \_$ so $C = n :: \_$

Fix a new $rev_2 : [a] \to [a]$ s.t.

$$n :: (rev_2\ xs) \triangleq rev_{+\!\!+}\ xs\ n$$

Expand $rev_{+\!\!+}$

$$=_\beta xs\begin{cases}[\,] \to n :: [\,] \\ (c :: cs) \to (rev_{+\!\!+}\ xs\ n) \mathbin{+\!\!+} [c]\end{cases}$$

Apply definition $n :: (rev_2\ xs) \triangleq rev_{+\!\!+}\ xs\ n$

$$= xs\begin{cases}[\,] \to n :: [\,] \\ (c :: cs) \to (n :: (rev_2\ xs)) \mathbin{+\!\!+} [c]\end{cases}$$

Expand ($+\!\!+$)

$$=_\beta xs\begin{cases}[\,] \to n :: [\,] \\ (c :: cs) \to n :: ((rev_2\ xs) \mathbin{+\!\!+} [c])\end{cases}$$

$(n :: \_)$ floats outside

$$= n :: \left(xs\begin{cases}[\,] \to [\,] \\ (c :: cs) \to (rev_2\ xs \mathbin{+\!\!+} [c])\end{cases}\right)$$

Cancel out $n :: \_$ to find $rev_2 =_\alpha rev$

$$rev_2\ xs = xs\begin{cases}[\,] \to [\,] \\ (c :: cs) \to (rev_2\ xs \mathbin{+\!\!+} [c])\end{cases}$$
$$=_\alpha rev\ xs$$

Function definitions

(List append) $$as \mathbin{+\!\!+} bs = as\begin{cases}[\,] \to bs \\ (c :: cs) \to c :: (cs \mathbin{+\!\!+} bs)\end{cases}$$

(List reversal) $$rev\ ds = ds\begin{cases}[\,] \to [\,] \\ (e :: es) \to (rev\ es) \mathbin{+\!\!+} [e]\end{cases}$$

(Natural number addition) $$x + y = x\begin{cases}0 \to y \\ s(z) \to s(z + y)\end{cases}$$

4. Deforesting $rev\ (rev\ xs)$

With factoring we can now calculate: $rev\ (rev\ xs) \leadsto xs$

Fix a new $id_r : [a] \to a \to [a]$, s.t.

$$id_r\ xs \triangleq rev\ (rev\ xs)$$

$$= T[\![rev\ (rev\ xs)]\!]$$

Expand rev

$$=_\beta T\left[\!\!\left[rev\ \left(xs\begin{cases}[\,] \to [\,] \\ (e :: es) \to (rev\ es) \mathbin{+\!\!+} [e]\end{cases}\right)\right]\!\!\right]$$

$T[\![rev\ (\_)]\!]$ floats inside

$$= xs\begin{cases}[\,] \to T[\![rev\ [\,]]\!] \\ (e :: es) \to T[\![rev\ ((rev\ es) \mathbin{+\!\!+} [e])]\!]\end{cases}$$

Apply deforestation

$$= xs\begin{cases}[\,] \to T[\![rev\ [\,]]\!] \\ (e :: es) \to T[\![rev_{+\!\!+}\ (rev\ es)\ e]\!]\end{cases}$$

Apply **factoring**

$$=_\alpha xs\begin{cases}[\,] \to T[\![rev\ [\,]]\!] \\ (e :: es) \to T[\![e :: (rev\ (rev\ es))]\!]\end{cases}$$

Apply definition $id_r\ xs \triangleq rev\ (rev\ xs)$

$$= xs\begin{cases}[\,] \to T[\![rev\ [\,]]\!] \\ (e :: es) \to T[\![e :: (id_r\ es)]\!]\end{cases}$$

Finish deforestation

$$=_\beta xs\begin{cases}[\,] \to [\,] \\ (e :: es) \to e :: (id_r\ es)\end{cases}$$

Factor result

$$= xs$$

5. Deforesting $x + x$

Factoring can also remove variable repetition, as in $x + x$

Fix a new $double\ x$ s.t.

$$double\ x \triangleq x + x$$

Expand $(+)$

$$= x\begin{cases}0 \to x \\ s(z) \to s(z + x)\end{cases}$$

Substitute pattern matched values for x

$$= x\begin{cases}0 \to 0 \\ s(z) \to s(z + s(z))\end{cases}$$

Factoring finds $z + s(z) \leadsto s(z + z)$

$$= x\begin{cases}0 \to 0 \\ s(z) \to s(s(z + z))\end{cases}$$

Apply definition $double\ x \triangleq x + x$

$$= x\begin{cases}0 \to 0 \\ s(z) \to s(s(double\ z))\end{cases}$$

$isEven(x + x)$, traditionally difficult, becomes trivial.

6. Results

Other results of just constant context factoring:

$$length\ (rev\ xs) \leadsto length\ xs$$
$$length\ (xs \mathbin{+\!\!+} xs) \leadsto double\ (length\ xs)$$

Results of our full method:

$$elem\ n\ (xs \mathbin{+\!\!+} [m]) \leadsto n = m \lor elem\ n\ xs$$
$$count\ n\ (insertsort\ xs) \leadsto count\ n\ xs$$
$$take\ (length\ xs)\ (rev\ xs) \leadsto rev\ xs$$
$$treesort\ xs \leadsto insertsort\ xs$$